**POTATO DISEASE CLASSIFICATION**

**USING CNN**

**A PROJECT REPORT**

*In partial fulfilment of the requirements for the award of the degree*

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**(DATA SCIENCE)**

*Under the guidance of*

**MAHENDRA DATTA**

**BY**

**ABHIRAM LAHA**

****

**FUTURE INSTITUTE OF ENGINEERING AND MANAGEMENT**

**In association with**



**(ISO9001:2015)**

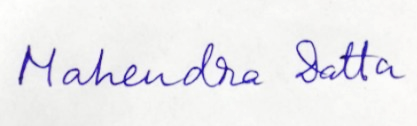
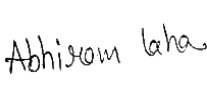
SDF Building, Module #132, Ground Floor, Salt Lake City, GP Block, Sector V, Kolkata, West Bengal 700091

***(Note: All entries of the proforma of approval should be filled up with appropriate and complete information. Incomplete proforma of approval in any respect will be summarily rejected.)***

|  |  |  |
| --- | --- | --- |
| 1. | Title of the Project: | **POTATO DISEASE CLASSIFICATION using CNN** |
| 2. | Project Member: | **ABHIRAM LAHA** |
| 3. | Name of the guide: | **Mr. MAHENDRA DUTTA** |
| 4. | Address: | Ardent Computech Pvt. Ltd  (An ISO 9001:2015 Certified)  SDF Building, Module #132, Ground Floor, Salt Lake City, GP Block, Sector V, Kolkata, West Bengal, 700091 |

***Project Version Control History***

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Primary Author** | **Description of Version** | **Date Completed** |
| Final | ABHIRAM LAHA | Project Report | 8TH MARCH,2024 |



Signature of Team Member Signature of Approver

Date: 08.03.2024 Date: 08.03.2024

For Office Use Only

**MR.MAHENDA DUTTA**

|  |
| --- |
| **Approved** |

|  |
| --- |
| **Not Approved** |

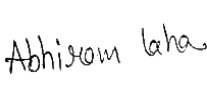
Project Proposal Evaluator

**DECLARATION**

We hereby declare that the project work being presented in the project proposal entitled **“POTATO DISEASE CLASSIFICATION using CNN”** in partial fulfilment of the requirements for the award of the degree of **BACHELOR OF** **TECHNOLOGY** at **ARDENT COMPUTECH PVT. LTD, SALTLAKE, KOLKATA, WEST BENGAL,** is an authentic work carried out under the guidance of **MR. MAHENDRA DUTTA**. The matter embodied in this project work has not been submitted elsewhere for the award of any degree of our knowledge and belief.

Date: 08.03.2024

Name of the Student: Abhiram Laha



***Signature of the student***



**Ardent Computech Pvt. Ltd (An ISO 9001:2015 Certified)**

SDF Building, Module #132, Ground Floor, Salt Lake City, GP Block, Sector V, Kolkata, West Bengal 700091

**CERTIFICATE**

This is to certify that this proposal of minor project entitled **“POTATO DISEASE CLASSIFICATION using CNN”** is a record of bonafide work, carried out by ***ABHIRAM LAHA*** under my guidance at **ARDENT** **COMPUTECH PVT LTD**. In my opinion, the report in its present form is in partial fulfilment ofthe requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** and as per regulations of the **ARDENT*®.*** To the best of my knowledge, the results embodied in this report, are original in nature and worthy of incorporation in the present version of the report.

**Guide / Supervisor**

------------------------------------------------

**MR. MAHENDRA DUTTA**

Project Engineer

Ardent Computech Pvt. Ltd (An ISO 9001:2015 Certified)

SDF Building, Module #132, Ground Floor, Salt Lake City, GP Block, Sector V, Kolkata, West Bengal 700091

**ACKNOWLEDGEMENT**

Success of any project depends largely on the encouragement and guidelines of many others. I take this sincere opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project work.

I would like to show our greatest appreciation to ***Mr. MAHENDRA DUTTA***, Project Engineer at Ardent, Kolkata. I always feel motivated and encouraged every time by his valuable advice and constant inspiration; without his encouragement and guidance this project would not have materialized.

Words are inadequate in offering our thanks to the other trainees, project assistants and other members at Ardent Computech Pvt. Ltd. for their encouragement and cooperation in carrying out this project work. The guidance and support received from all the members and who are contributing to this project, was vital for the success of this project.

**CONTENTS**

* Overview
* History of Python
* Environment Setup
* Basic Syntax
* Variable Types
* Functions
* Modules
* Packages
* Machine Learning
  + Supervised and Unsupervised Learning
  + NumPy
  + Pandas
  + Matplotlib
* Deep Learning
* What is Deep Learning?
* Artificial Neural Networks
* Convolutional Neural Networks
* Tensorflow
* POTATO DISEASE CLASSIFICATION

1. Introduction
2. Problem Statement
3. Objective
4. Actual Codes For Potato Disease classification
5. Advantages & Disadvantages
6. Conclusion
7. Future Scope
8. Bibliography

**OVERVIEW**

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and has fewer syntactical constructions than other languages.

**Python is interpreted**: Python is processed at runtime by the interpreter. You do not need to compileyour program before executing it. This is similar to Perl and PHP.

**Python is Interactive**: You can actually sit at a Python prompt and interact with the interpreterdirectly to write your programs.

**Python is Object-Oriented**: Python supports Object-Oriented style or technique of programmingthat encapsulates code within objects.

**Python is a Beginner's Language**: Python is a great language for the beginner-level programmersand supports the development of a wide range of applications from simple text processing to WWW browsers to games.

**HISTORY OF PYTHON**

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands. Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Small Talk, UNIX shell, and other scripting languages. Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL). Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

FEATURES OF PYTHON

Easy-to-learn: Python has few Keywords, simple structure and clearly defined syntax. This allows a student to pick up the language quickly.

Easy-to-Read: Python code is more clearly defined and visible to the eyes.

Easy -to-Maintain: Python's source code is fairly easy-to-maintain.

A broad standard library: Python's bulk of the library is very portable and cross platform compatible on UNIX, Windows, and Macintosh.

Interactive Mode: Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

Portable: Python can run on the wide variety of hardware platforms and has the same interface on all platforms.

Extendable: You can add low level modules to the python interpreter. These modules enables programmers to add to or customize their tools to be more efficient.

Databases: Python provides interfaces to all major commercial databases.

GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries, and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

Scalable: Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below:

* It support functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to byte code for building

large applications.

* It provides very high level dynamic datatypes and supports dynamic type checking.
* It supports automatic garbage collections.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA and JAVA.

**ENVIRONMENT SETUP**

Open a terminal window and type "python" to find out if it is already installed and which version is installed.

* UNIX (Solaris, Linux, FreeBSD, AIX, HP/UX, SunOS, IRIX, etc.)
* Win 9x/NT/2000
* Macintosh (Intel, PPC, 68K)
* OS/2
* DOS (multiple versions)
* PalmOS
* Nokia mobile phones
* Windows CE
* Acorn/RISC OS

**BASIC SYNTAX OF PYTHON PROGRAM**

Type the following text at the Python prompt and press the Enter –

>>> print "Hello, Python!"

*If you are running new version of Python, then you would need to use print statement with parenthesis* as in **print ("Hello, Python!");**.

However in Python version 2.4.3, this produces the following result –

Hello, Python!

Python Identifiers

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (\_) followed by zero or more letters, underscores and digits (0 to 9).

Python does not allow punctuation characters such as @, $, and % within identifiers. Python is a case sensitive programming language.

Python Keywords

The following list shows the Python keywords. These are reserved words and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only.

**And, exec, not**

**Assert, finally, or**

**Break, for, pass**

**Class, from, print**

**continue, global, raise**

**def, if, return**

**del, import, try**

**elif, in, while**

**else, is, with**

**except, lambda, yield**

Lines & Indentation

Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced.

The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example –

if True:

print "True"

else:

print "False"

Command Line Arguments

Many programs can be run to provide you with some basic information about how they should be run. Python enables you to do this with -h −

$ python-h

usage: python [option]...[-c cmd|-m mod | file |-][arg]...

Options and arguments (and corresponding environment variables):

-c cmd: program passed in as string(terminates option list)

-d : debug output from parser (also PYTHONDEBUG=x)

-E : ignore environment variables (such as PYTHONPATH)

-h : print this help message and exit [ etc.]

**VARIABLE TYPES**

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

Assigning Values to Variables

Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable. The equal sign (=) is used to assign values to variables.

counter=10 # An integer assignment

weight=10.60 # A floating point

name="Ardent" # A string

Multiple Assignment

Python allows you to assign a single value to several variables simultaneously. For example −

a = b = c = 1

a,b,c = 1,2,"hello"

Standard Data Types

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has five standard data types −

 String

 List

 Tuple

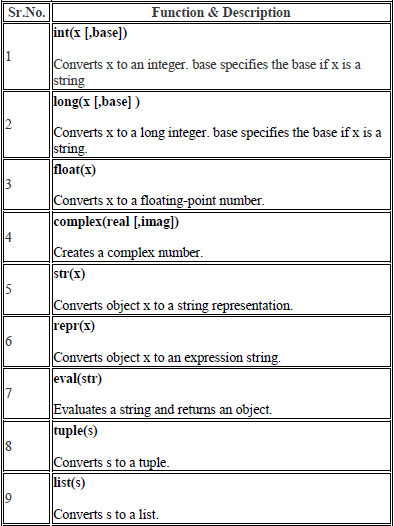
 Dictionary

 Number

Data Type Conversion

Sometimes, you may need to perform conversions between the built-in types. To convert between types, you simply use the type name as a function.

There are several built-in functions to perform conversion from one data type to another.



**FUNCTIONS**

Defining a Function

* def function name( parameters ):

"function\_docstring"

function suite

return [expression]

Pass by reference vs Pass by value

All parameters (arguments) in the Python language are passed by reference. It means if you change what a parameter refers to within a function, the change also reflects back in the calling function. For example –

*# Function definition is here*

def change me(mylist):

"This changes a passed list into this function"

mylist.append([1,2,3,4]);

print"Values inside the function: ",mylist

return

*# Now you can call changeme function*

mylist=[10,20,30];

change me(mylist);

print” Values outside the function: ",mylist

Here, we are maintaining reference of the passed object and appending values in the same object. So, this would produce the following result −

Values inside the function: [10, 20, 30, [1, 2, 3, 4]]

Values outside the function: [10, 20, 30, [1, 2, 3, 4]]

Global vs. Local variables

Variables that are defined inside a function body have a local scope, and those defined outside have a global scope . For Example-

total=0; # This is global variable.

*# Function definition is here*

def sum( arg1, arg2 ):

*# Add both the parameters and return them."*

total= arg1 + arg2; # Here total is local variable.

print"Inside the function local total: ", total

return total;

*# Now you can call sum function*

sum(10,20);

Print”Outside the function global total: ", total

When the above code is executed, it produces the following result −

Inside the function local total: 30

Outside the function global total: 0

**MODULES**

A module allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. A module is a Python object with arbitrarily named attributes that you can bind and reference.

The Python code for a module named *aname* normally resides in a file named *aname.py*. Here's an example of a simple module, support.py

def print\_func( par ):

print"Hello : ", par

return

The *import* Statement

You can use any Python source file as a module by executing an import statement in some other Python source file. The *import* has the following syntax –

Import module1 [, module2 [… moduleN]

**PACKAGES**

A package is a hierarchical file directory structure that defines a single Python application environment that consists of modules and sub packages and sub-subpackages, and so on.

Consider a file *Pots.py* available in *Phone* directory. This file has following line of source code −

def Pots ():

print "I'm Pots Phone"

Similar way, we have another two files having different functions with the same name as above –

 *Phone/Isdn.py* file having function Isdn ()

 *Phone/G3.py* file having function G3 ()

Now, create one more file \_\_init\_\_.py in *Phone* directory −

 Phone/\_\_init\_\_.py

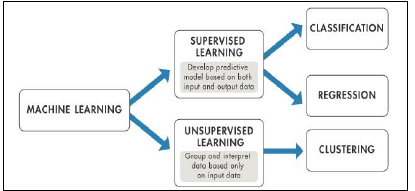
To make all of your functions available when you've imported Phone, you need to put explicit import statements in \_\_init\_\_.py as follows −

from Pots import Pots

from Isdn import Isdn

from G3 import

**MACHINE LEARNING**



Machine learning is a field of computer science that gives computers the ability to learn without being explicitly programmed.

Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data.

**INTRODUCTION TO MACHINE LEARNING**

**Machine learning** is a field of computer science that gives computers the ability to learn without being explicitly programmed.

**Arthur Samuel**, an American pioneer in the field of computer gaming and artificial intelligence, coined the term "Machine Learning" in 1959 while at IBM. Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data

Machine learning tasks are typically classified into two broad categories, depending on whether there is a learning "signal" or "feedback" available to a learning system:-

SUPERVISED LEARNING

**Supervised learning** is the machine learning task of inferring a function from *labelled training data*.[1] The training data consist of a set of *training examples*. In supervised learning, each example is a *pair* consisting of an input object (typically a vector) and a desired output value.

A supervised learning algorithm analyses the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way.

UNSUPERVISED LEARNING

**Unsupervised learning** is the machine learning task of inferring a function to describe hidden structure from "unlabelled" data (a classification or categorization is not included in the observations). Since the examples given to the learner are unlabelled, there is no evaluation of the accuracy of the structure that is output by the relevant algorithm—which is one way of distinguishing unsupervised learning from supervised learning and reinforcement learning.

A central case of unsupervised learning is the problem of density estimation in statistics, though unsupervised learning encompasses many other problems (and solutions) involving summarizing and explaining key features of the data.

NUMPY

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin.

NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents.

Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted, and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars.

NUMPY ARRAY

NumPy’s main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In NumPy dimensions are called *axes*. The number of axes is *rank*.

For example, the coordinates of a point in 3D space [1, 2, 1] is an array of rank 1, because it has one axis. That axis has a length of 3. In the example pictured below, the array has rank 2 (it is 2-dimensional). The first dimension (axis) has a length of 2, the second dimension has a length of 3.

[[1., 0., 0.],

[ 0., 1., 2.]]

NumPy’s array class is called *ndarray*. It is also known by the alias.

SLICING NUMPY ARRAY

**Import numpy as np**

**a = np.array ([[1, 2, 3],[3,4,5],[4,5,6]])**

print 'Our array is:'

Print a

print '\n'

print 'The items in the second column are:'

print a[...,1]

print '\n'

print 'The items in the second row are:'

print a[1...]

print '\n'

print 'The items columns 1 onwards are:'

print a [...,1:]

**OUTPUT**

Our array is:

[[1 2 3]

[3 4 5]

[4 5 6]]

The items in the second column are:

[2 4 5]

The items in the second row are:

[3 4 5]

The items column 1 onwards are:

[[2 3]

[4 5]

[5 6]]

MATPLOTLIB

**Matplotlib** is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter,wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged .SciPy makes use of matplotlib.

**EXAMPLE**

* **LINE PLOT**

>>>importmatplotlib.pyplotasplt

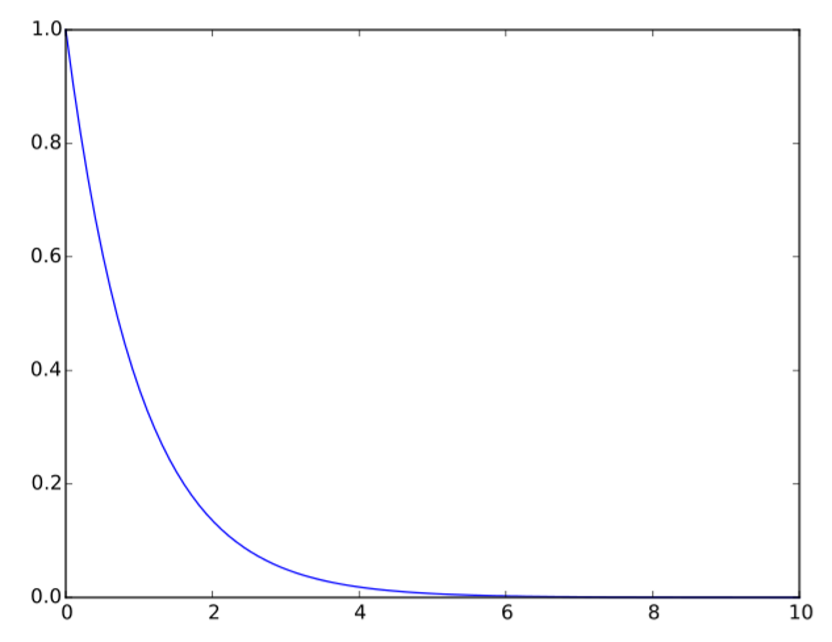
>>>importnumpyasnp

>>> a =np.linspace(0,10,100)

>>> b =np.exp(-a)

>>>plt.plot (a,b)

>>>plt.show ()



* **SCATTER PLOT**

>>>importmatplotlib.pyplotasplt

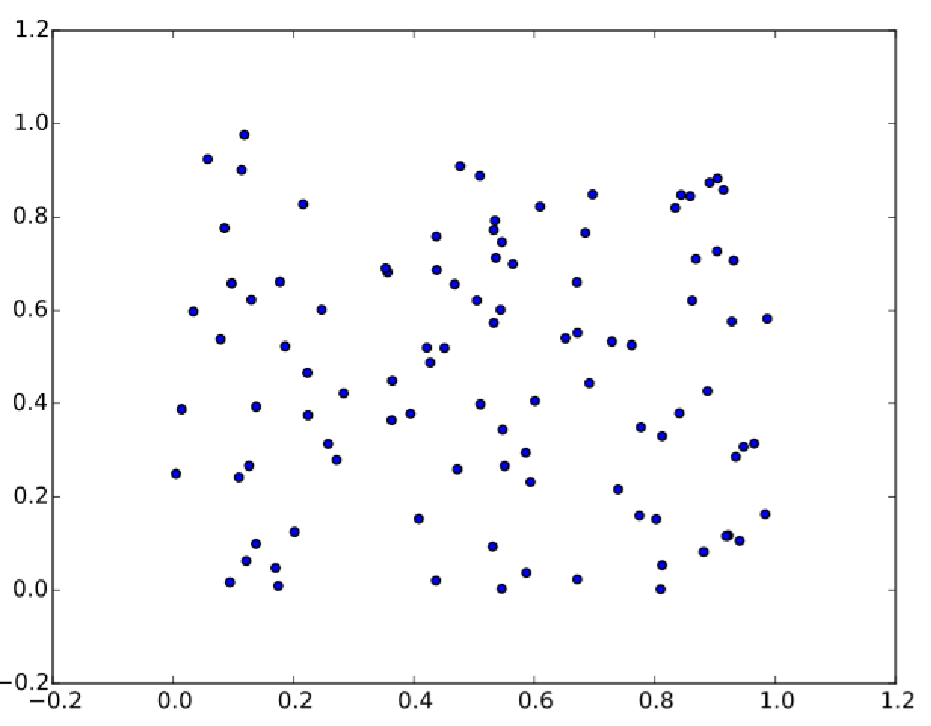
>>>fromnumpy.randomimport rand

>>> a =rand(100)

>>> b =rand(100)

>>>plt.scatter(a, b)

>>>plt.show ()



PANDAS

In computer programming, **pandas** is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. "Panel data", an econometrics term for multidimensional, structured data sets.

**LIBRARY FEATURES**

* Data Frame object for data manipulation with integrated indexing.
* Tools for reading and writing data between in-memory data structures and different file formats.
* Data alignment and integrated handling of missing data.
* Reshaping and pivoting of data sets.
* Label-based slicing, fancy indexing, and sub setting of large data sets.
* Data structure column insertion and deletion.
* Group by engine allowing split-apply-combine operations on data sets.
* Data set merging and joining.
* Hierarchical axis indexing to work with high-dimensional data in a lower-dimensional data structure.
* Time series-functionality: Date range generation.

**DEEP LEARNING**

WHAT IS DEEP LEARNING?

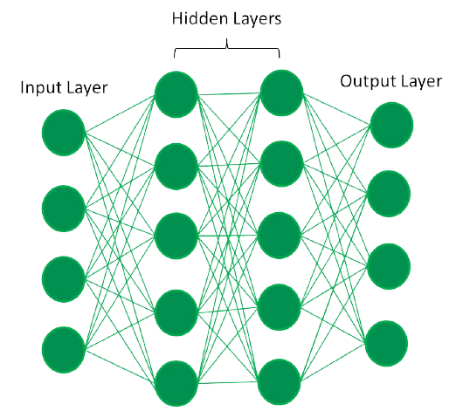
Deep learning is the branch of [machine learning](https://www.geeksforgeeks.org/machine-learning/) which is based on artificial neural network architecture. An artificial neural network or ANN uses layers of interconnected nodes called neurons that work together to process and learn from the input data.

In a fully connected Deep neural network, there is an input layer and one or more hidden layers connected one after the other. Each neuron receives input from the previous layer neurons or the input layer. The output of one neuron becomes the input to other neurons in the next layer of the network, and this process continues until the final layer produces the output of the network. The layers of the neural network transform the input data through a series of nonlinear transformations, allowing the network to learn complex representations of the input data.

ARTIFICIAL NEURAL NETWORKS

[Artificial neural networks](https://www.geeksforgeeks.org/artificial-neural-networks-and-its-applications/) are built on the principles of the structure and operation of human neurons. It is also known as neural networks or neural nets. An artificial neural network’s input layer, which is the first layer, receives input from external sources and passes it on to the hidden layer, which is the second layer. Each neuron in the hidden layer gets information from the neurons in the previous layer, computes the weighted total, and then transfers it to the neurons in the next layer.

These connections are weighted, which means that the impacts of the inputs from the preceding layer are more or less optimized by giving each input a distinct weight. These weights are then adjusted during the training process to enhance the performance of the model.

****

In a fully connected artificial neural network, there is an input layer and one or more hidden layers connected one after the other. Each neuron receives input from the previous layer neurons or the input layer. The output of one neuron becomes the input to other neurons in the next layer of the network, and this process continues until the final layer produces the output of the network. Then, after passing through one or more hidden layers, this data is transformed into valuable data for the output layer. Finally, the output layer provides an output in the form of an artificial neural network’s response to the data that comes in.

CONVOLUTIONAL NEURAL NETWORK

In deep learning, a **convolutional neural network** ([**CNN**](https://www.analyticsvidhya.com/blog/2018/12/guide-convolutional-neural-network-cnn/)**/ConvNet**) is a class of deep neural networks, most commonly applied to analyze visual imagery. Now when we think of a neural network we think about matrix multiplications but that is not the case with ConvNet. It uses a special technique called Convolution. Now in mathematics **convolution** is a mathematical operation on two functions that produces a third function that expresses how the shape of one is modified by the other.



Convolutional neural networks are composed of multiple layers of artificial neurons.

TensorFlow

TensorFlow is an open-source machine learning framework developed by Google Brain Team. It provides a comprehensive ecosystem of tools, libraries, and community resources that allow researchers and developers to build and deploy machine learning models effectively. TensorFlow supports various tasks in machine learning, including neural networks, deep learning, reinforcement learning, and more.

Key features of TensorFlow include:

* **Flexible Architecture**: TensorFlow offers a flexible architecture that allows developers to deploy computations across multiple CPUs, GPUs, or TPUs (Tensor Processing Units).
* **High-level APIs**: TensorFlow provides high-level APIs like Keras, which simplifies the process of building and training deep learning models. Keras is integrated directly into TensorFlow, making it easy to use and efficient.
* **Scalability**: TensorFlow enables scalable deployment of machine learning models, making it suitable for both small-scale and large-scale applications.

**POTATO DISEASE CLASSIFICATION USING CNN**

**INTRODUCTION**

CNNs are particularly well-suited for image classification tasks like this because they can automatically learn hierarchical features from image data. Classifying potato diseases using Convolutional Neural Networks (CNNs) is a fascinating application of deep learning in agriculture.

It's essential to have a sufficiently large and diverse dataset, as well as careful pre-processing and tuning of the CNN architecture, to achieve accurate and robust classification of potato diseases. Additionally, collaboration with domain experts, such as agronomists or plant pathologists, can provide valuable insights into the specific characteristics of each disease and aid in dataset curation and model interpretation.

Here's a general introduction to how we approached this task:

* Data collection and Pre-processing: Collect a diverse dataset of potato disease images. Ensure that the dataset covers various types of diseases, stages of infection, and environmental conditions.
* Model Architecture: Design a CNN architecture suitable for image classification. This typically consists of convolutional layers followed by pooling layers to extract features from the images.
* Training: Split the dataset into training, validation, and testing sets. Train the CNN model using the training data. During training, the model learns to classify images into different disease categories by adjusting its parameters based on the provided labels.
* Evaluation: Evaluate the trained model on the testing set to assess its performance in classifying unseen data.
* Deployment: Once satisfied with the model's performance, deploy it for practical use. This could involve integrating it into a mobile or web application, a drone-based monitoring system, or a smart farming platform.

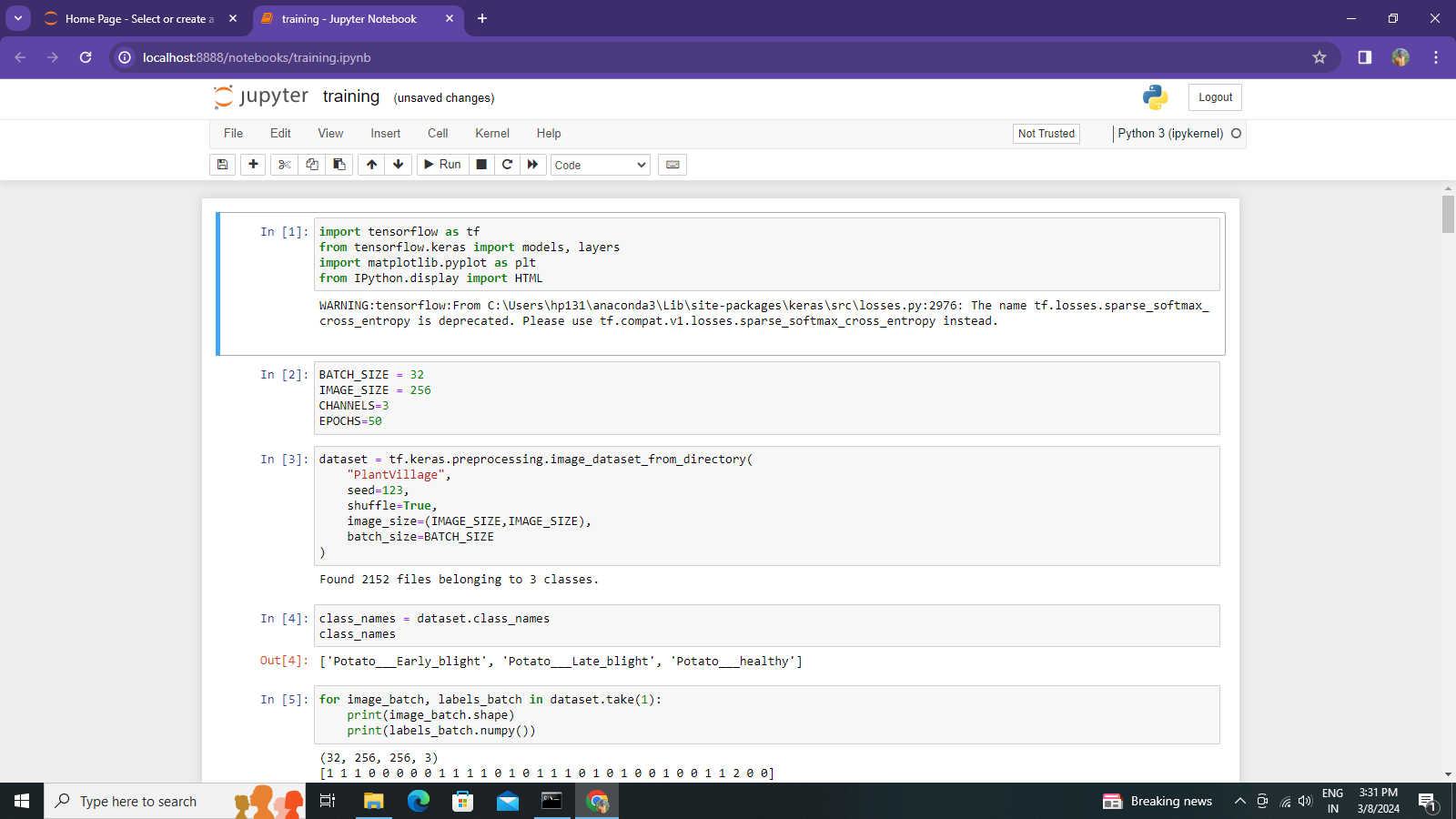
**PROBLEM STATEMENT**

Developing an accurate and efficient deep learning model to classify various types of potato diseases from images, with the aim of assisting farmers and agronomists in timely detection and management of plant health issues. The model should be capable of distinguishing between healthy potato plants and those affected by common diseases, including but not limited to late blight, early blight, potato virus Y, and bacterial wilt.

**OBJECTIVE**

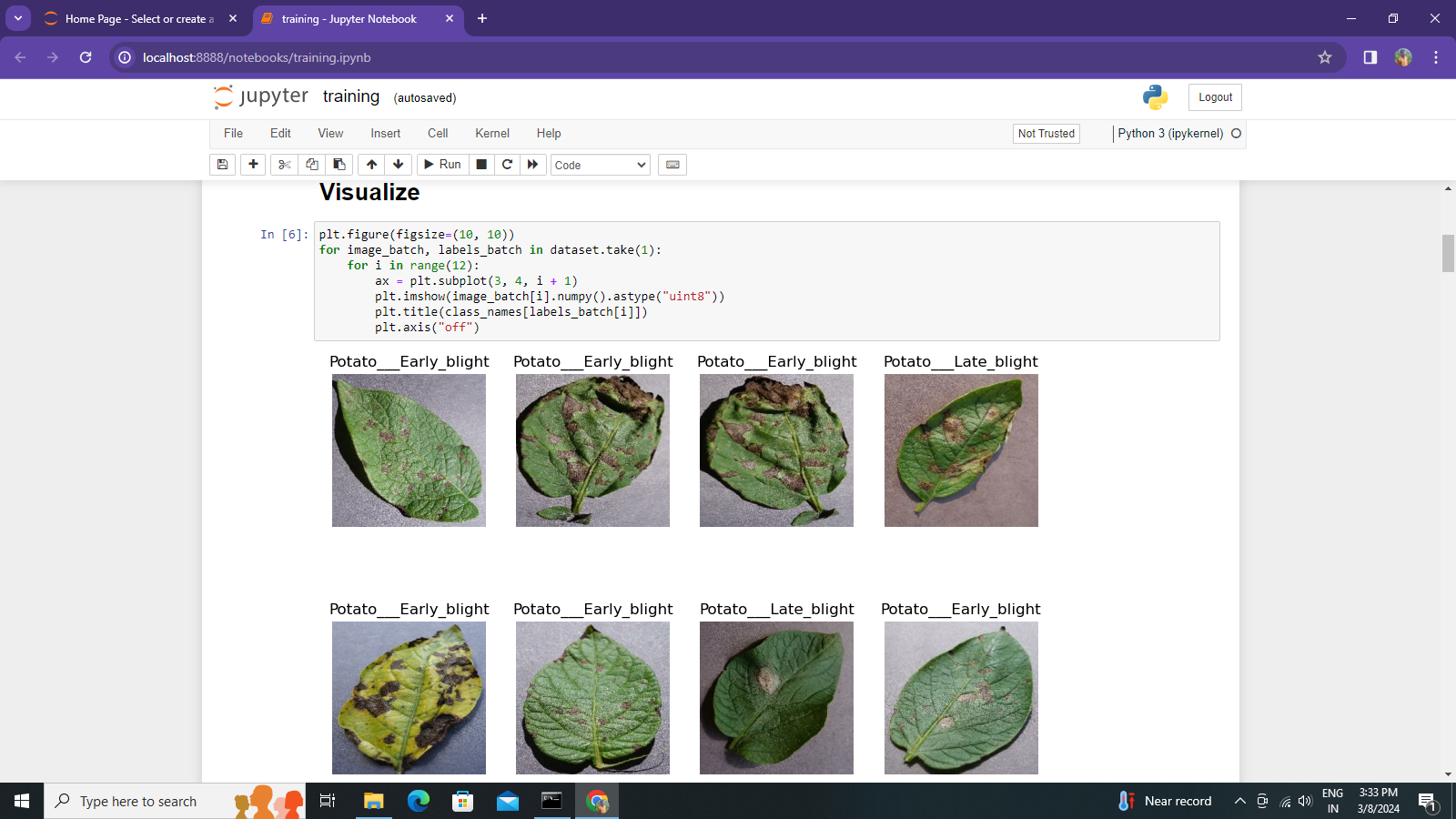
The objective is to develop a robust Convolutional Neural Network (CNN) model for accurately classifying potato plant images into healthy and diseased categories. This involves collecting and pre-processing a diverse dataset, optimizing model training, and evaluating performance metrics. The ultimate aim is to deploy the model into practical agricultural applications, integrating seamlessly with existing systems. Continuous monitoring and iteration ensure the model's effectiveness in aiding farmers and agronomists in timely disease detection and management, thereby enhancing crop yield and sustainability.

**ACTUAL CODES FOR POTATO DISEASE CLASSIFICATION USING CNN**

****

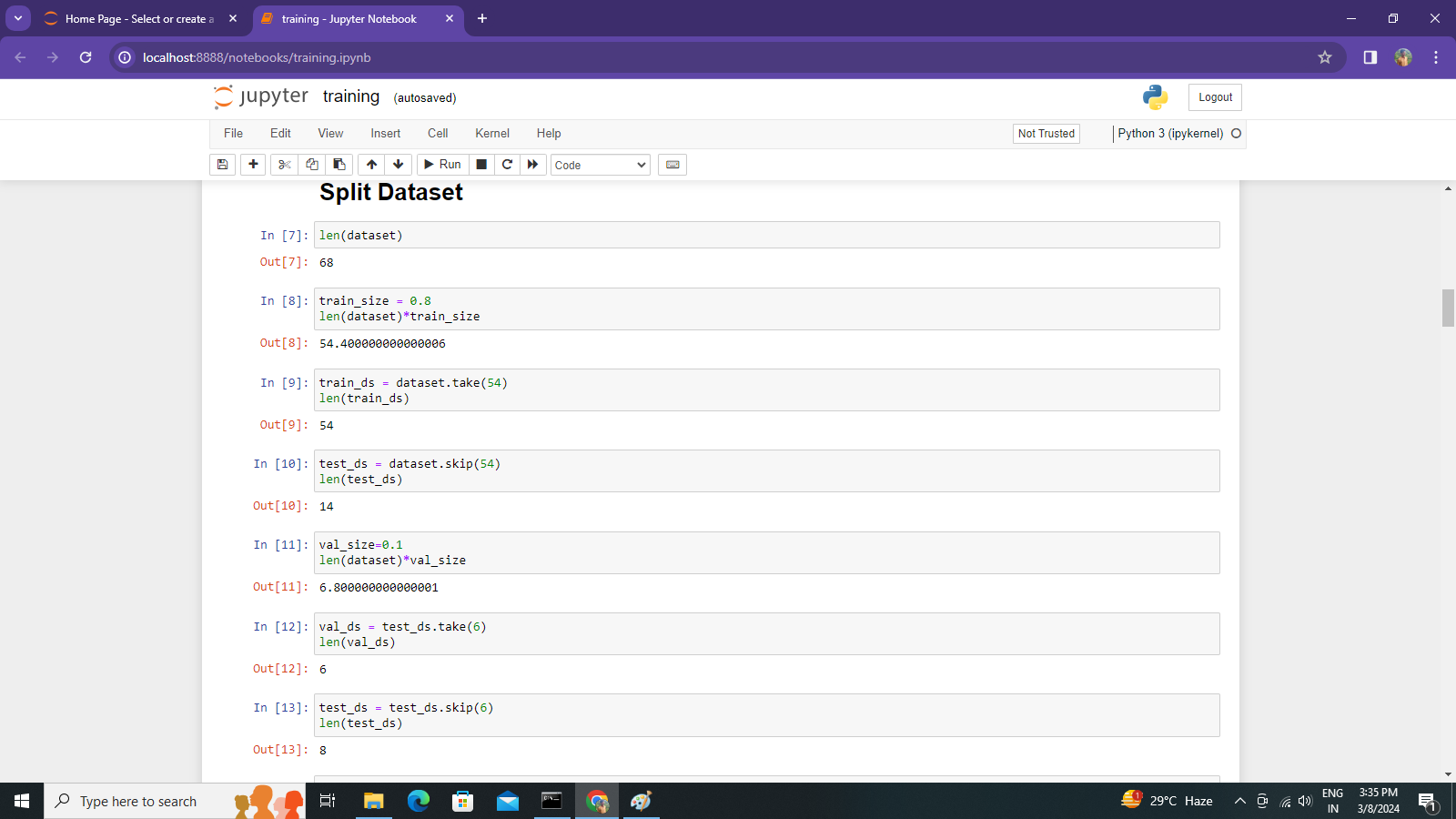
**Explanation :**

First we need to import the TensorFlow ,its libraries, matplotlib. This code snippet imports necessary libraries such as TensorFlow and Matplotlib for building and visualizing models. It assumes the existence of a dataset variable. It retrieves and prints the class names from the dataset. Then, it iterates through the dataset, printing the shape of image batches and their corresponding labels.



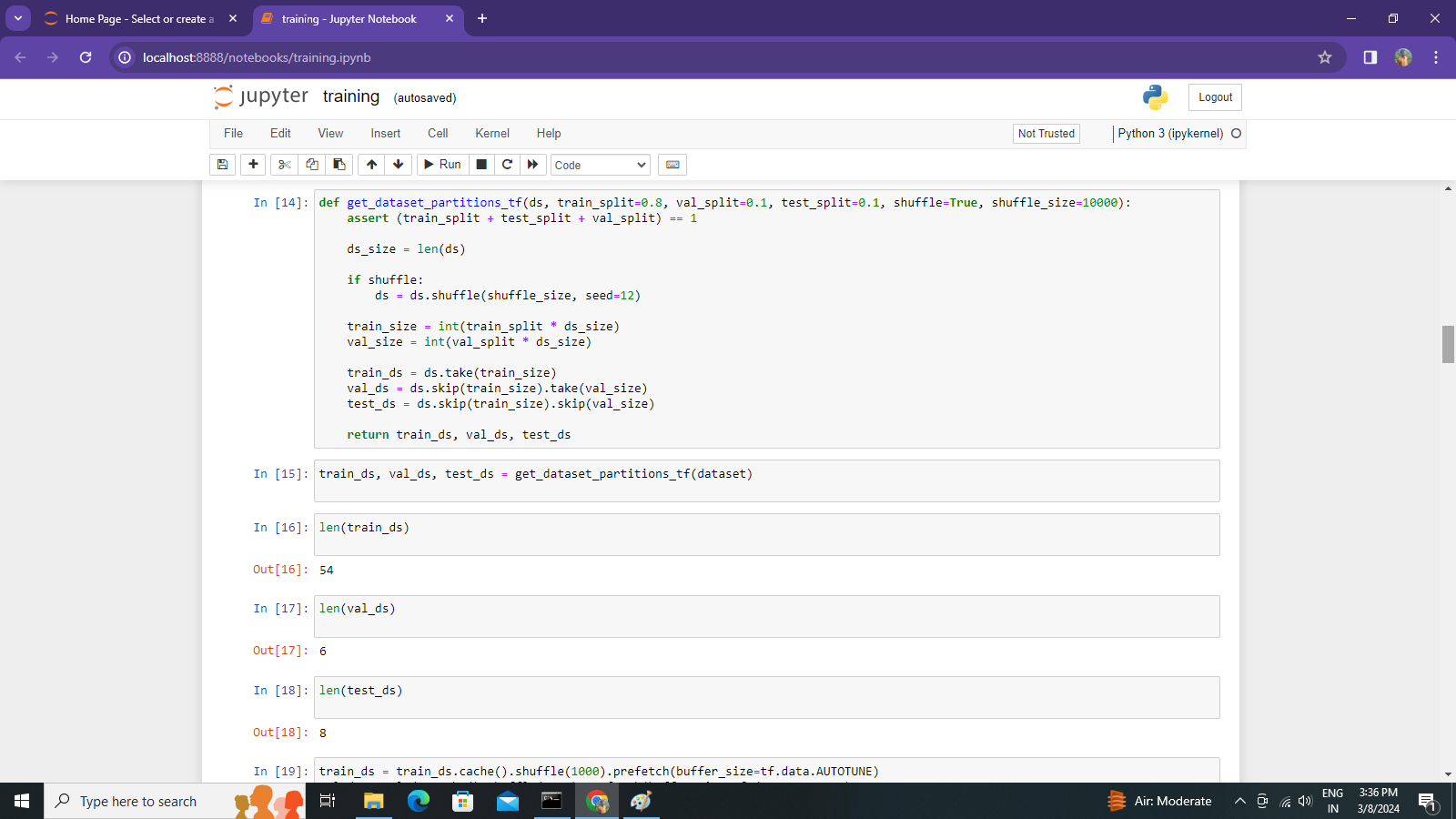
**Explanation :**

This code snippet creates a figure for plotting images from the dataset. It iterates through the dataset, taking one batch of images and labels. Within this loop, it iterates over a subset of images in the batch, plotting each image along with its corresponding label. The subplot layout is organized in a 3x4 grid. Each image is displayed with its label as the title, and axes are turned off for cleaner visualization.



**Explanation :**

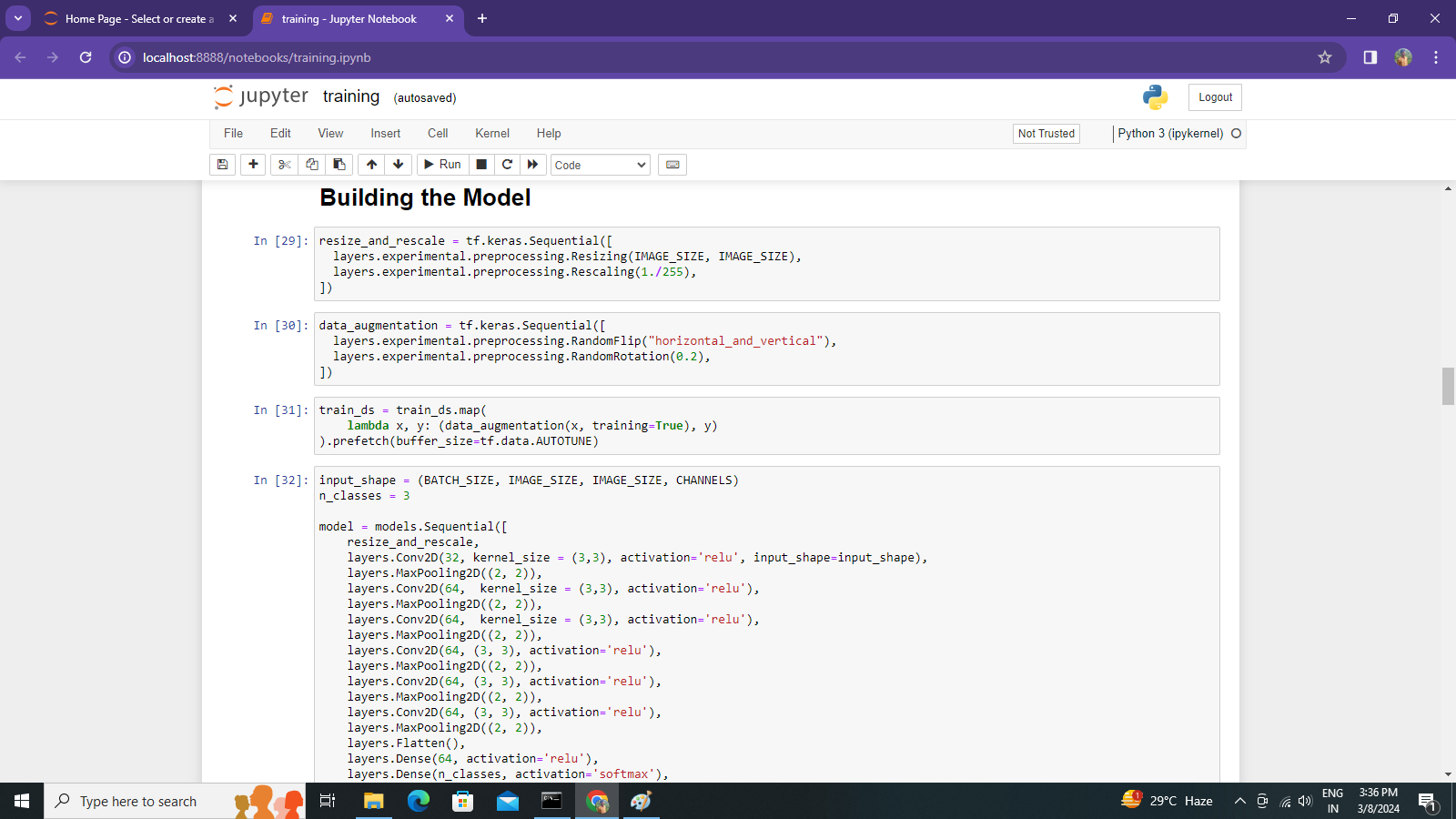
This code snippet splits the dataset into training, validation, and test sets using specific proportions. It calculates the size of the training and validation sets based on percentages of the total dataset. Then, it creates the training set by taking the first portion of the dataset, the test set by skipping the training set, and finally, it splits a portion of the test set for validation and assigns the rest for testing. Each set is assigned to a respective variable and the lengths of each set are calculated accordingly.



**Explanation :**

This function **get\_dataset\_partitions\_tf** takes a dataset (**ds**) and splits it into training, validation, and test sets according to specified proportions. It also allows shuffling of the dataset before splitting. The function ensures that the sum of the split proportions is equal to 1.

After that code uses the **get\_dataset\_partitions\_tf** function to split the dataset into training, validation, and test sets. It assigns the returned values from the function to the variables **train\_ds**, **val\_ds**, and **test\_ds**, respectively, allowing further processing or training of machine learning models on each partition of the dataset.



**Explanation :**

This code defines a convolutional neural network (CNN) model for classifying images of potato diseases

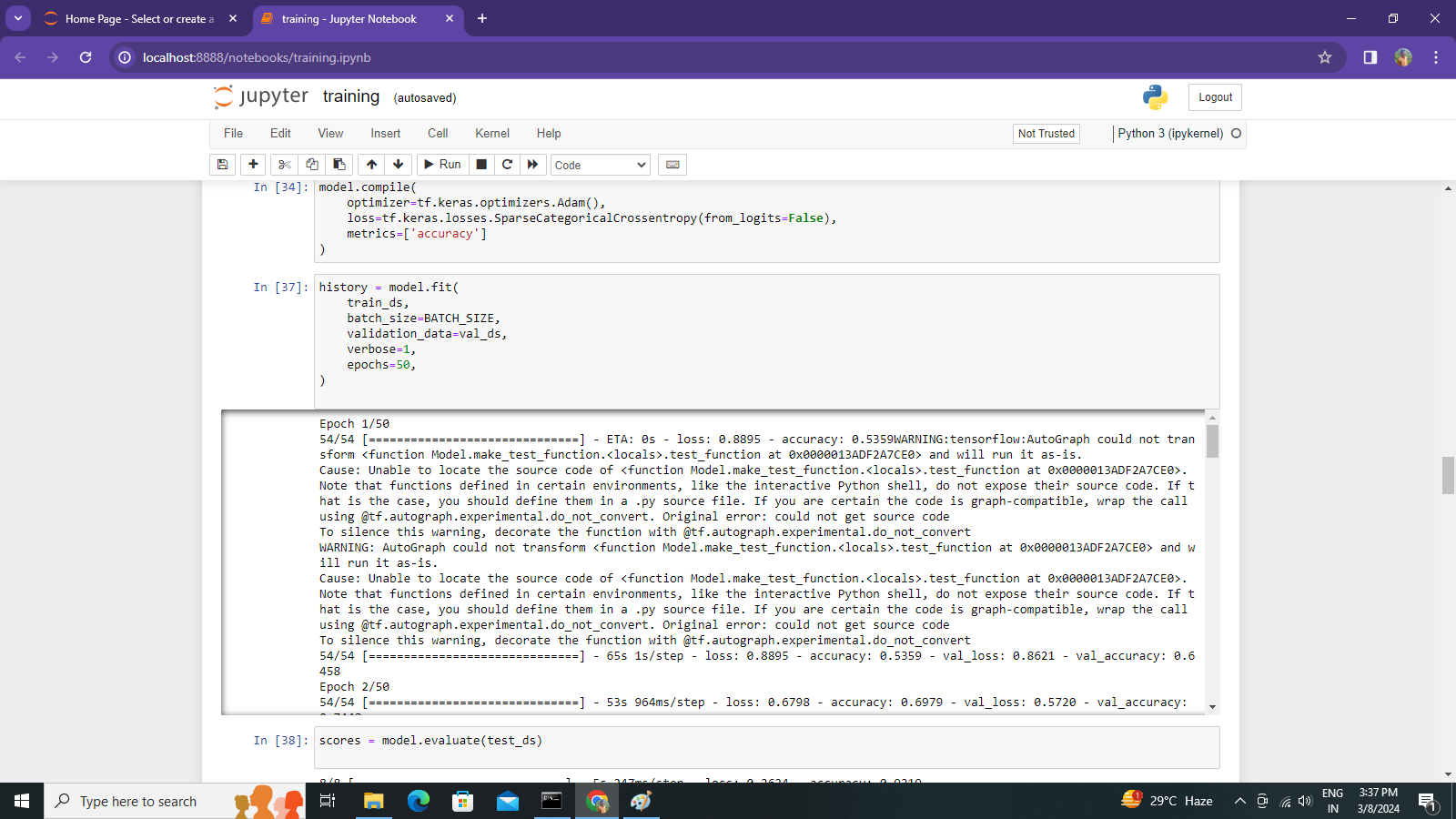
**resize\_and\_rescale**: Sequential model composed of image preprocessing layers. It resizes input images to a specified size (**IMAGE\_SIZE**) and then rescales pixel values to the range [0,1] by dividing by 255.

**data\_augmentation**: Sequential model for data augmentation, which randomly flips images horizontally and vertically and applies random rotations up to 20%.

**train\_ds.map(...)**: Applies data augmentation to the training dataset using the **map** function, ensuring it's done in batches and prefetching data for optimization.

**input\_shape**, **n\_classes**: Variables defining the input shape of images, batch size (**BATCH\_SIZE**), image size (**IMAGE\_SIZE**), number of channels (**CHANNELS**), and number of classes.

**model**: Sequential model composed of convolutional and pooling layers, followed by dense layers for classification. It stacks multiple Conv2D layers with ReLU activation and MaxPooling2D layers to extract features from input images. The final layers consist of Flatten, Dense (with ReLU activation), and Dense (with softmax activation for multiclass classification) layers.

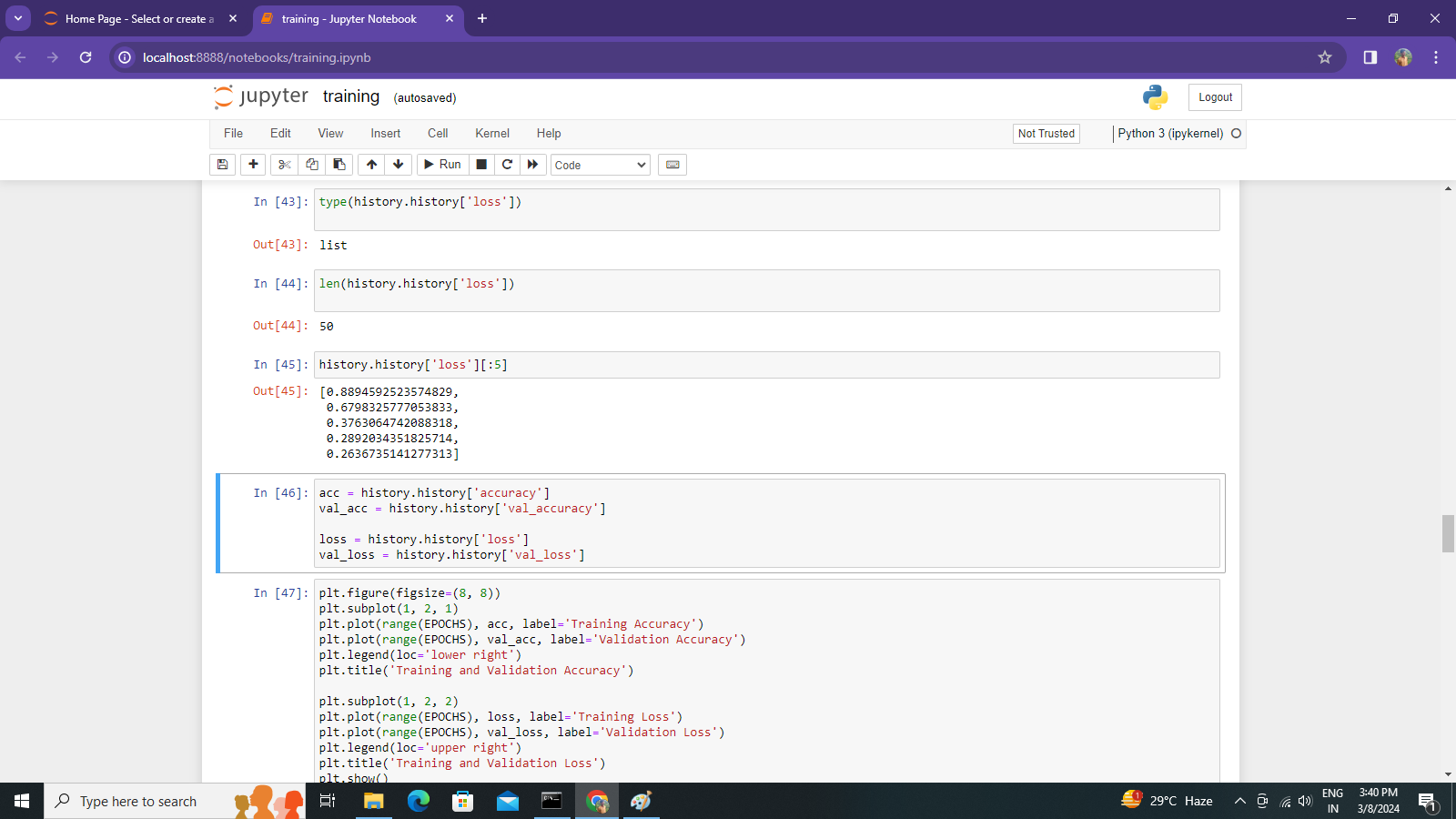


**Explanation :**

**model.compile**: Configures the model for training, specifying the optimizer, loss function, and evaluation metrics. In this case, it uses the Adam optimizer, sparse categorical crossentropy loss (suitable for integer labels), and accuracy as the metric to monitor.

**model.fit**: Trains the model on the training dataset (**train\_ds**) for a specified number of epochs (50 in this case), with a given batch size. It also specifies the validation dataset (**val\_ds**) for monitoring the model's performance during training. The **verbose** parameter controls the verbosity mode during training.

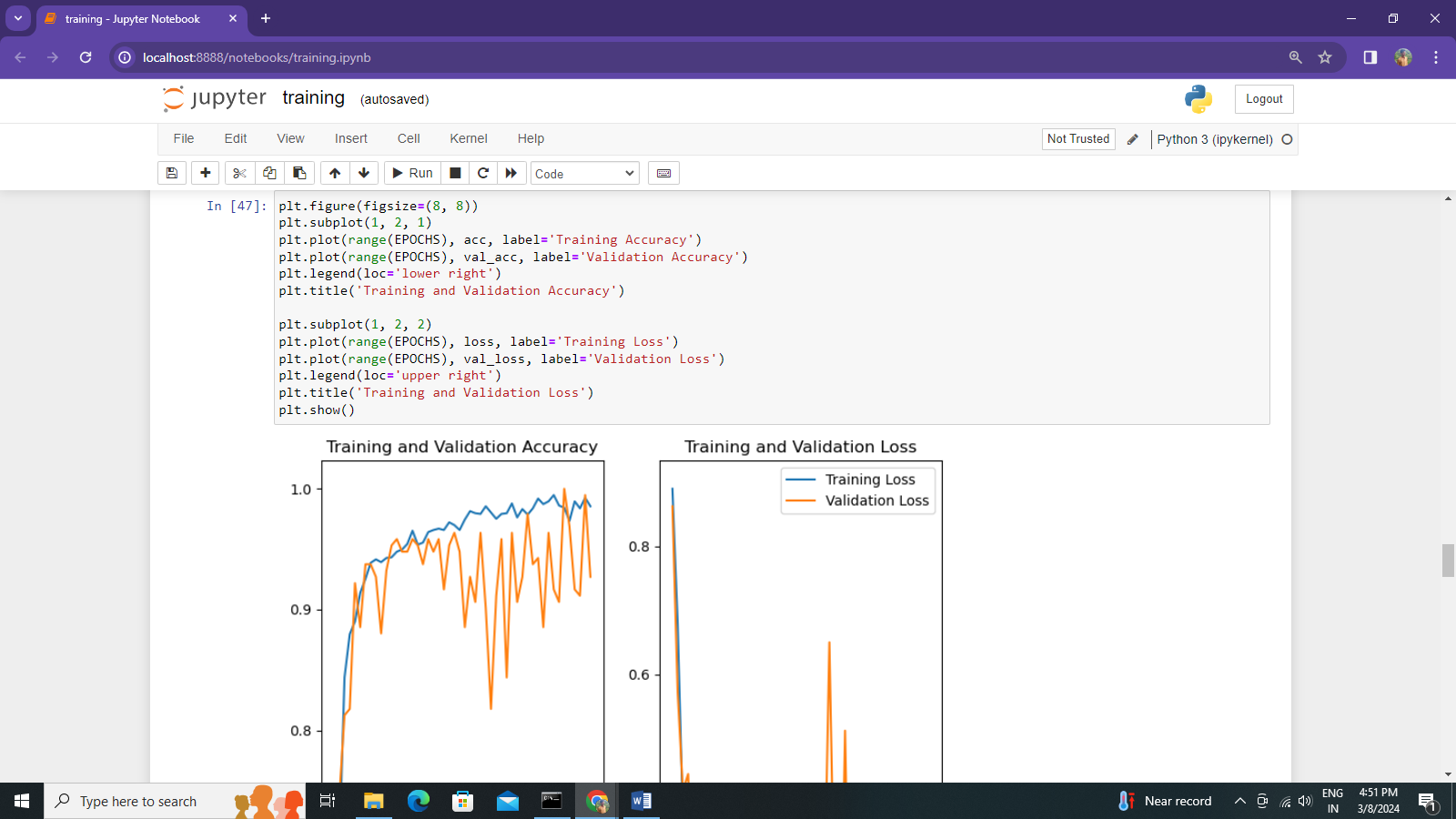
**history**: Stores the training history, including loss and accuracy values, which can be used for visualization and analysis.



**Explanation :**

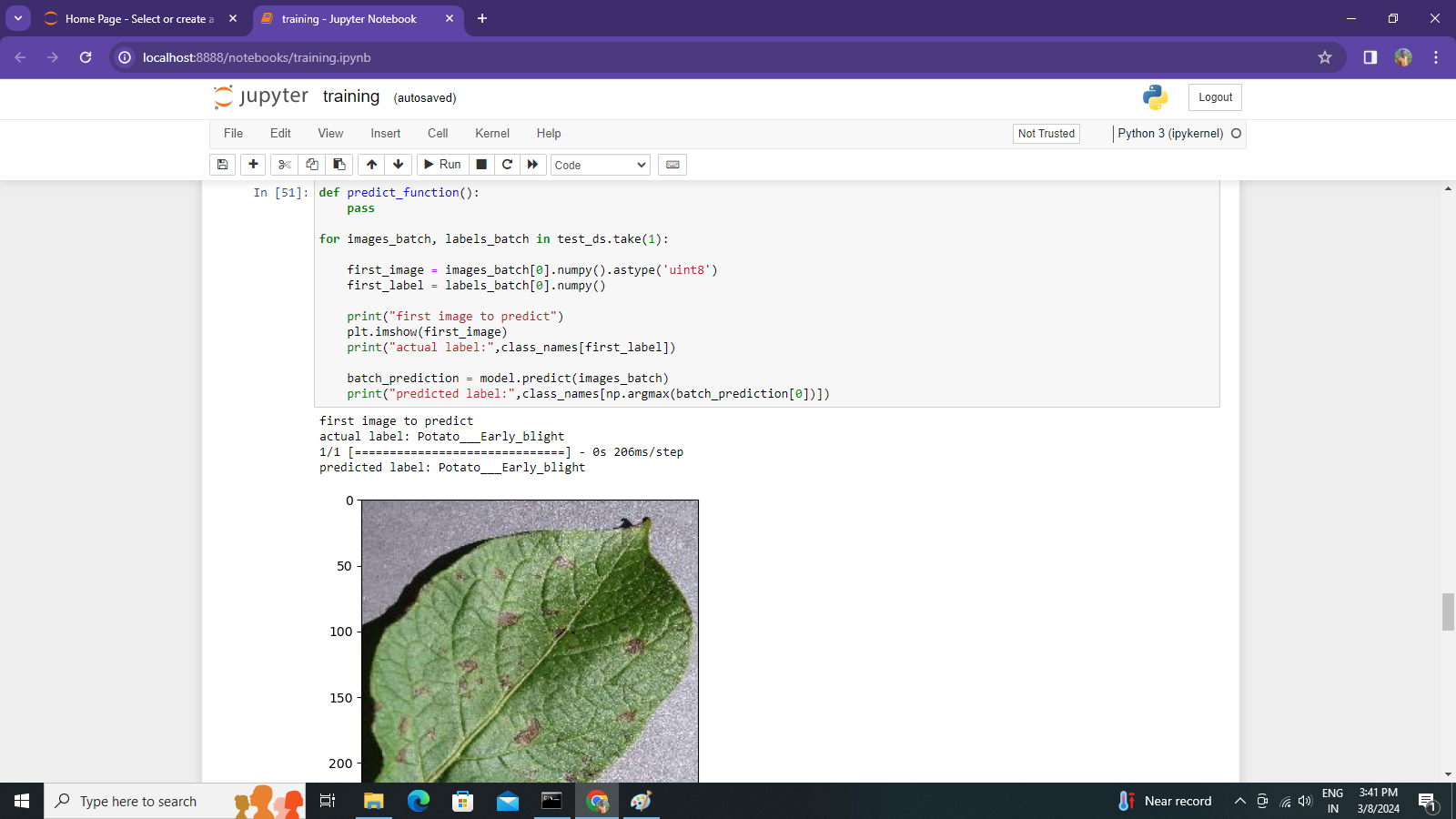
These lines of code extract the accuracy and loss values from the training history (**history**) dictionary, which was obtained from the training process using **model.fit**.

These values can be used for plotting training and validation curves to analyze the model's performance over the training epochs. Typically, accuracy values are plotted against epochs to visualize the model's learning progress, while loss values are plotted to monitor how well the model is optimizing its parameters



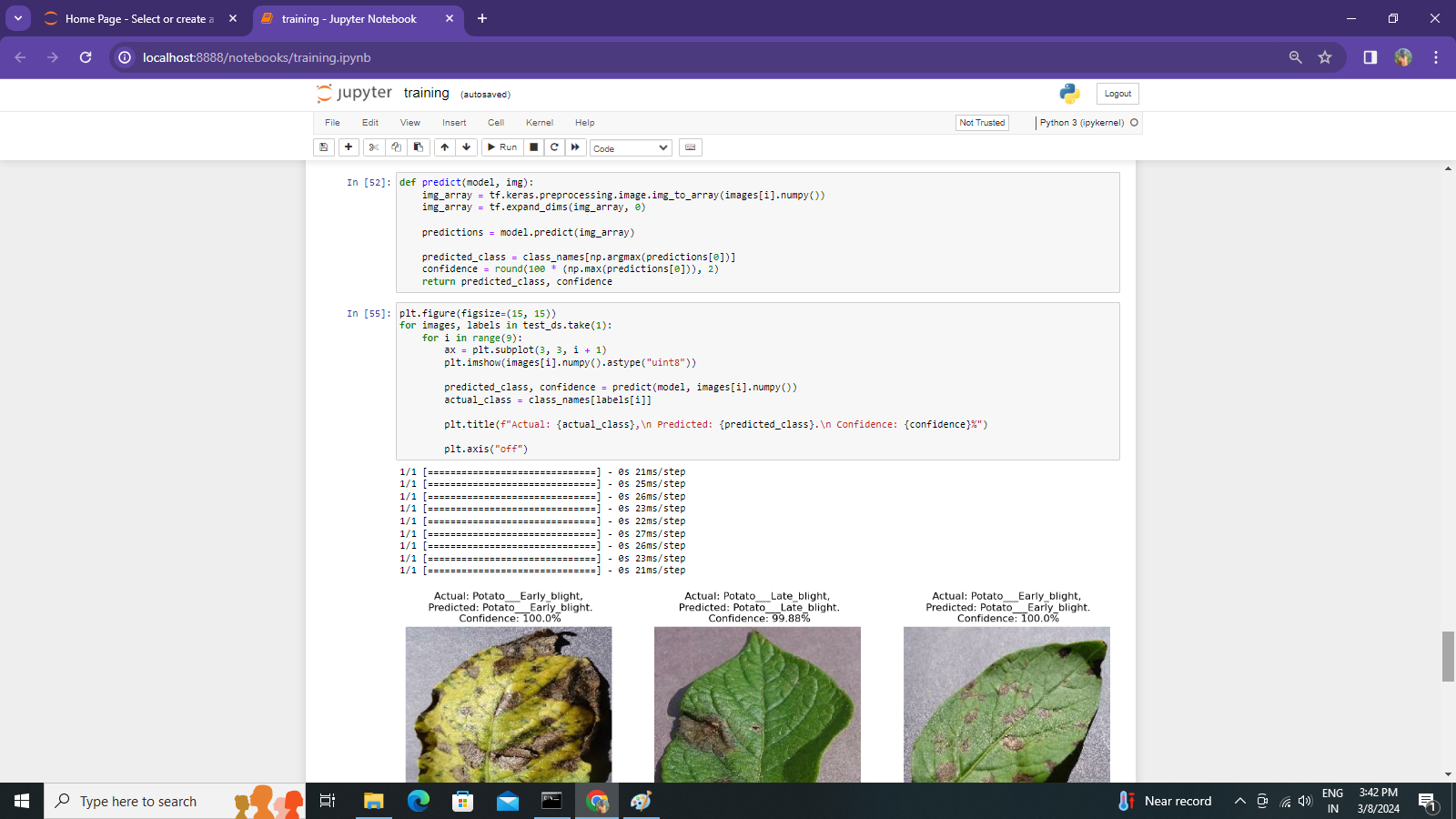
**Explanation :**

This code uses Matplotlib to create two subplots within a single figure, showing the training and validation accuracy on one subplot and the training and validation loss on the other. It plots these metrics against the range of epochs. Legends and titles are included for clarity, and the figure is displayed using **plt.show()**.



**Explanation :**

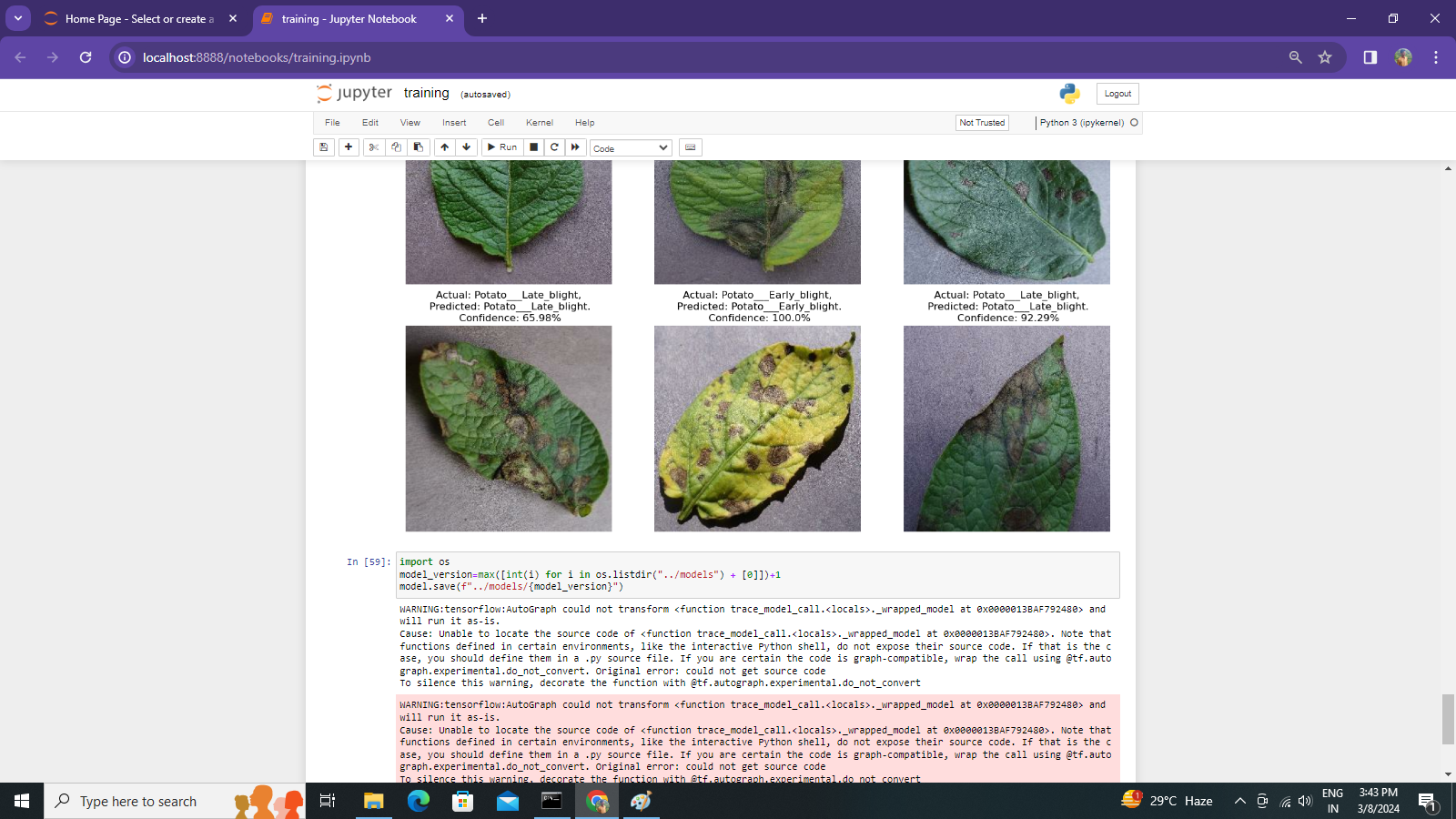
This code snippet initializes a function named **predict\_function**, which is currently empty. It then proceeds to iterate through a single batch of images and their associated labels from the test dataset (**test\_ds**). For each image in the batch, it retrieves the first image and its corresponding label. It prints the actual label of the first image and utilizes the trained model to predict labels for all images in the batch. Subsequently, it prints the predicted label for the first image. Finally, it displays the first image using Matplotlib for visual reference.



**Explanation :**

This **predict** function serves to predict the class and confidence score of an input image using a provided trained model. It begins by converting the input image into a NumPy array using **tf.keras.preprocessing.image.img\_to\_array()**. The array is then expanded to match the expected input shape of the model. Subsequently, the model is used to predict the class probabilities for the input image. The predicted class is determined by selecting the class with the highest probability from the prediction results, and the confidence score is calculated as the maximum probability multiplied by 100, rounded to two decimal places. Finally, the function returns the predicted class label and confidence score.

The code in line 18 generates a 3x3 grid of images from the test dataset (**test\_ds**) and their corresponding predicted and actual class labels with confidence scores. It iterates through a batch of images and labels, displaying each image along with its predicted and actual labels using the **predict** function defined earlier. The predicted class and confidence are obtained by calling the **predict** function with the trained model and the current image. The title of each subplot includes the actual and predicted class labels, along with the confidence score in percentage. The figure size is set to 15x15 inches for better visualization, and the axes are turned off for cleaner presentation.



**Explanation :**

This code snippet determines the latest version number for a saved model by finding the maximum integer value among the directories in the "../models" directory and incrementing it by 1. It then saves the current model to a new directory with the calculated version number. This approach helps maintain a versioned history of saved models, ensuring easy management and retrieval.

model.save("../potatoes.h5")

This line of code saves the trained model to a file named "potatoes.h5" in the specified directory.

**ADVANTAGES**

1. Increased accuracy for effective heart

disease diagnosis.

2. Handles roughest(enormous) amount of

data using random forest algorithm and

feature selection.

3. Reduce the time complexity of doctors.

4. Cost effective for patients.

Classifying potato diseases using machine learning techniques like CNNs offers several advantages:

* **Early Disease Detection**: By accurately identifying diseases in potato plants at an early stage, farmers can take proactive measures to prevent further spread, leading to better crop health and higher yields.
* **Reduced Labor Intensity**: Automated disease classification systems can streamline the monitoring process, reducing the need for manual inspection and enabling more efficient resource allocation on farms.
* **Crop Protection :** Preventing or mitigating disease outbreaks through early detection and intervention helps safeguard potato crops, contributing to food security and stability in regions reliant on potato cultivation.

**DISADVANTAGES**

1. Prediction of cardiovascular disease

results is not accurate.

* **Data Limitations**: Obtaining a comprehensive dataset of potato disease images that adequately represents all possible diseases, stages of infection, and environmental conditions can be challenging. Limited or biased data may result in models that fail to generalize well to unseen conditions.
* **Model Complexity**: Developing and training accurate machine learning models, such as CNNs, requires expertise in data preprocessing, model architecture design, hyperparameter tuning, and computational resources. Complex models may be difficult to interpret and maintain.

**CONCLUSION**

Utilizing machine learning techniques for potato disease classification offers numerous benefits such as early disease detection, precision agriculture, and improved yield, it also presents several challenges. Issues such as data limitations, model complexity, and resource intensiveness can hinder the development and deployment of effective classification systems. Moreover, concerns regarding model robustness, ethical considerations, and the potential for false positives or negatives underscore the importance of careful validation and interpretation of results. Despite these challenges, addressing them through collaborative efforts involving researchers, farmers, policymakers, and technologists can lead to the development of robust and ethically responsible solutions that enhance potato crop management and contribute to global food security. As technology continues to evolve, it is essential to prioritize the development of sustainable, equitable, and reliable machine learning approaches for potato disease classification, thereby fostering innovation while addressing the diverse needs and challenges of agricultural communities worldwide.

**FUTURE SCOPE**

The future scope of potato disease classification using machine learning techniques is promising and encompasses several avenues for further development and application:

* **Real-time Monitoring and Decision Support**: Advancements in sensor technologies, internet-of-things (IoT) devices, and cloud computing infrastructure enable real-time monitoring of potato crops.
* **Enhanced Accuracy and Robustness**: Continued research and innovation in machine learning algorithms, data collection methods, and model training techniques can lead to more accurate and robust classification systems.

**BIBLIOGRAPHY**

[1] Used kaggle dataset for training and testing https://www.kaggle.com/code/amankrpandey1/potato-disease-classification

[2] Took help from the website. https://www.tensorflow.org/tutorials/images/classification

[3] Help from an Article https://potatoes.ahdb.org.uk/knowledge-library/potato-disease-identification

**THANK YOU**